FIAP ON

DESENVOLVIMENTO DE APPS -
PARTE 1 (ANDROID)

INTEGRAÇÃO COM WEBSERVICES & API

ROBERTO RODRIGUES JÚNIOR

7

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

7 INTEGRAÇÃO COM WEBSERVICES & API

Os principais aplicativos usados hoje em dia fazem uso de recursos da nuvem, sejam serviços, banco de dados, entre outros. *Facebook, WhatsApp, Twitter e Instagram* são exemplos de aplicativos que não existiriam sem o uso de *webservices*. As aplicações dinâmicas permitem que um aplicativo converse com outros sistemas por meio dos *webservices.*

7.1 Consumindo informações com a biblioteca Retrofit

A *Retrofit* é uma *API* desenvolvida pela *Square* seguindo padrão *REST*, fornecendo um padrão simples de implementação para transmissão de dados entre aplicação e servidor, que faz uso do *JSON.* Este capítulo tem por finalidade a apresentação da *API Retrofit* de conexão *HTTP* para *Android*. O entendimento sobre como a *API Retrofit* pode ser utilizada é importante, porque é considerada uma ótima opção quando há necessidade de integração entre sua aplicação e serviços disponibilizados em servidores remotos ou conhecidos como *backend*.

A *API Retrofit* é composta pelo uso de outras *APIs* para garantir aquilo que promove de melhor: a conexão e o consumo de *webservices* padrão RESTful. A empresa *Square* utilizou a seguinte estrutura:

1. *API* padrão Android/Kotlin para executar requisições;
2. A biblioteca *GSON* para serialização das respostas *JSON*;
3. A biblioteca *OkHttp* para tratar de conectar, efetuar cache e fazer chamadas *HTTP* mantidas pela mesma *Square.*

O *Retrofit* tornou mais fácil a recuperação e o envio de *JSON* por meio de uma *webservice REST*. Existem dezenas de *webservices* gratuitos com as mais diferentes informações que podem ser consumidas, entre eles, destacamos:

- **Clima e Temperatura**: https://openweathermap.org/api
  Personagens Marvel: https://developer.marvel.com/
  Filmes: http://www.omdbapi.com/
- **“Que a força esteja com você”**: https://swapi.dev

- **Informações sobre CEP**: https://viacep.com.br/

Escolhemos a *API* que fornece informações de endereços por meio do CEP da viaCEP. Acompanhe o desenvolvimento das rotinas em Kotlin, em conjunto com a biblioteca *Retrofit,* que acessaram as informações em um aplicativo *Android.* Abra o *Android Studio* e siga as instruções da Figura “Android Studio Novo Projeto”:

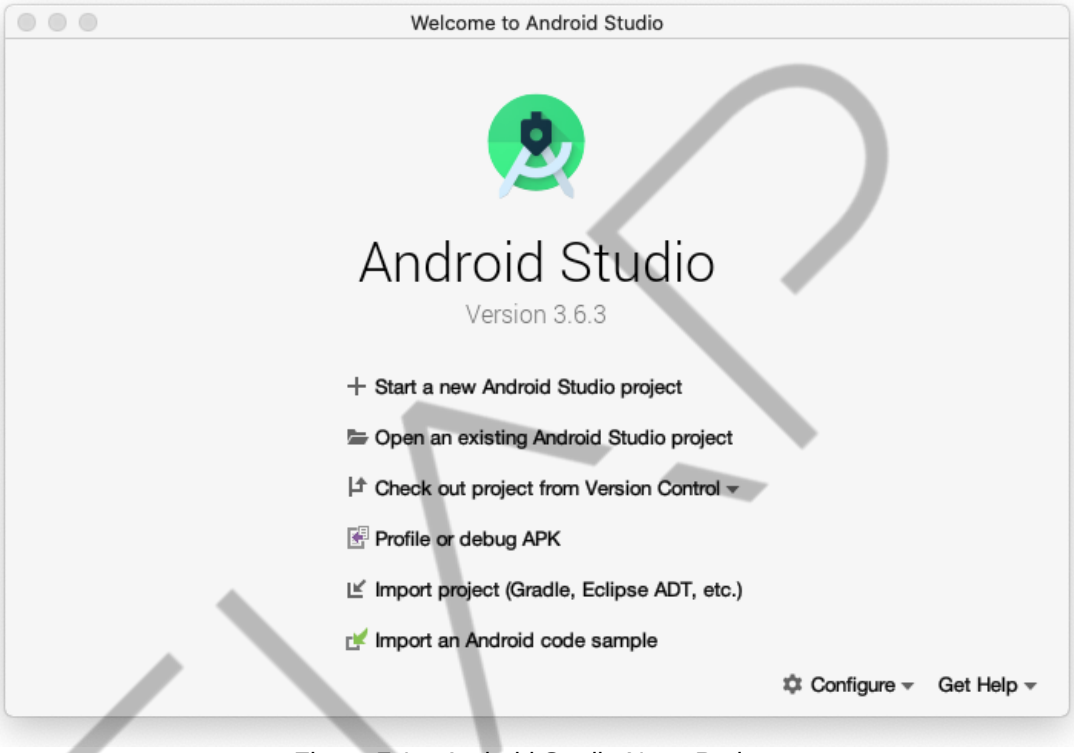

Figura 7.1 – Android Studio Novo Projeto
Fonte: Elaborado pelo autor (2020)

Na tela de templates de projeto, selecione a opção “Empty Activity”, como mostra a Figura “Android Studio Template de Projeto”.

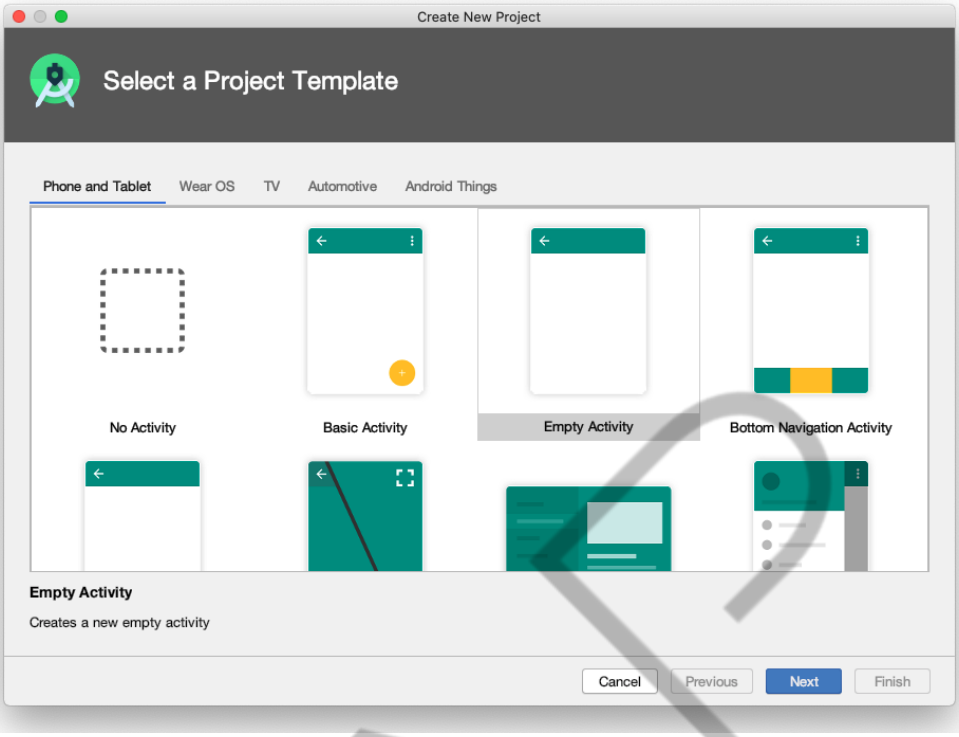

Figura 7.2 – Android Studio Template de Projeto
Fonte: Elaborado pelo autor (2020)

O nome do projeto será "*WebService*", escolha Kotlin como a linguagem de programação e complete os dados conforme a Figura "Android Studio Detalhes do Projeto". Desenvolveremos o aplicativo para celulares Android a partir da versão 7.0 (Nougat) conhecida como API 24.

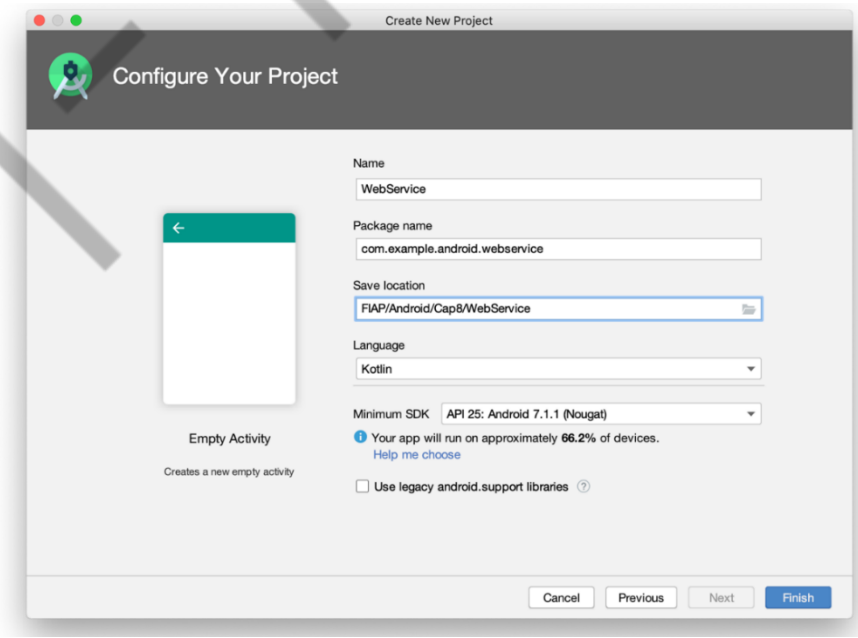

Figura 7.3 – Android Studio Detalhes do Projeto
Fonte: Elaborado pelo autor (2020)

Após clicar no botão *Finish*, aguarde o processamento de construção do projeto, depois disso, siga as próximas orientações para desenvolver e configurar os arquivos necessários para acessar o *webservice* proposto neste capítulo. Adicione no arquivo *build.gradle* na sessão "*dependencies*" as dependências mostradas no Código-fonte “Dependências Retrofit e Gson”. E, por último, mas não menos importante, clique no botão "*Sync Now*" no lado direito superior, para atualizar as nossas dependências:

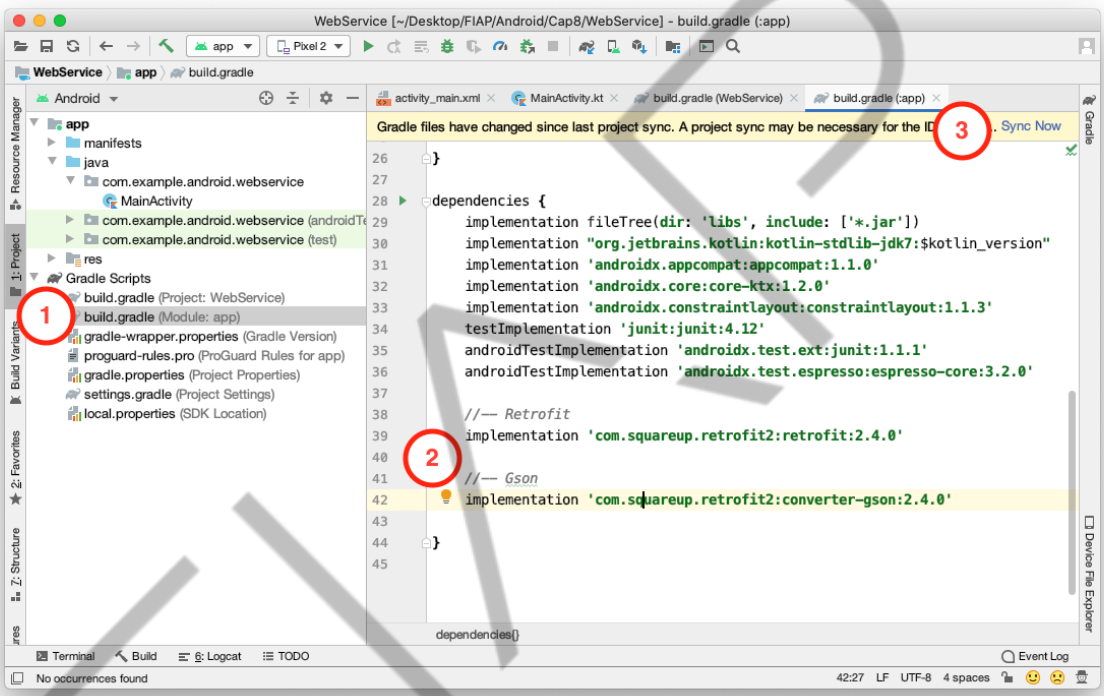

Figura 7.4 – Android Studio Dependências Retrofit e Gson
Fonte: Elaborado pelo autor (2020)

```
//-- Retrofit
implementation 'com.squareup.retrofit2:retrofit:2.4.0'

//-- Gson
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
```

Código-fonte 7.1 – Dependências Retrofit e Gson
Fonte: Elaborado pelo autor (2018)

Crie uma classe dos dados chamada CEP com o conteúdo do Código-fonte “Classe de Dados CEP”:

```
import com.google.gson.annotations.SerializedName
```

```
data class CEP(

        @SerializedName("cep")          val cep: String,
        @SerializedName("logradouro")   val logradouro: String,
        @SerializedName("complemento")  val complemento: String,
        @SerializedName("bairro")       val bairro: String,
        @SerializedName("localidade")   val localidade: String,
        @SerializedName("uf")           val uf: String,
        @SerializedName("unidade")      val unidade: String,
        @SerializedName("ibge")         val ibge: String,
        @SerializedName("gia")          val gia: String
)
```

Código-fonte 7.2 – Classe de Dados CEP
Fonte: Elaborado pelo autor (2018)

Crie a interface *RetrofitService.kt* e classe *RetrofitFactory.kt*.

No RetrofitService são criadas as assinaturas de funções para o serviço a ser consumido. O RetrofitFactory gera o objeto de conexão para consumo do serviço.

```
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Path

interface RetrofitService {

    //https://viacep.com.br/ws/01538001/json/
    @GET("{CEP}/json/")
    fun getCEP(@Path("CEP") CEP : String) : Call<CEP>


//https://viacep.com.br/ws/SP/Sao%20Paulo/Avenida%20Lins%20de%20Vasconcel
os/json/
    @GET("{estado}/{cidade}/{endereco}/json/")
    fun getRCE(@Path("estado") estado: String,
               @Path("cidade") cidade: String,
               @Path("endereco") endereco: String): Call<List<CEP>>
}
```

Código-fonte 7.3 – Classe RetrofitService
Fonte: Elaborado pelo autor (2018)

```
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class RetrofitFactory {

    val URL: String = "https://viacep.com.br/ws/"

    val retrofitFactory = Retrofit.Builder()
            .baseUrl(URL)
```

```
            .addConverterFactory(GsonConverterFactory.create())
            .build()

    fun retrofitService(): RetrofitService {
        return retrofitFactory.create(RetrofitService::class.java)
    }
}
```

Código-fonte 7.4 – Classe RetrofitFactory
Fonte: Elaborado pelo autor (2018)

Altere o arquivo chamado "*MainActivity.kt*" com as linhas indicadas:

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //-- Ao clicar no botão número 1
        //-- Será pesquisado o logradouro com o número do CEP
        pesquisaCEP.setOnClickListener {

            progress_bar.visibility = View.VISIBLE

            val call =
RetrofitFactory().retrofitService().getCEP(cep.text.toString())

            call.enqueue(object : Callback<CEP> {

                override fun onResponse(call: Call<CEP>, response:
Response<CEP>) {

                    response.body()?.let {
                        Log.i("CEP", it.toString())
                        Toast.makeText(this@MainActivity,
it.toString(), Toast.LENGTH_LONG).show()
                        progress_bar.visibility = View.INVISIBLE
                    } ?: Toast.makeText(this@MainActivity, "CEP não
localizado", Toast.LENGTH_LONG)
                        .show()

                }

                override fun onFailure(call: Call<CEP>?, t:
Throwable?) {
```

```
                    Log.e("Erro", t?.message)
                    progress_bar.visibility = View.INVISIBLE
                }
            })
        }

        //-- Ao clicar no botão número 2
        //-- Será pesquisado o logradouro com os dados:
        //-- RUA, CIDADE e ENDEREÇO
        pesquisaRCE.setOnClickListener {

            progress_bar.visibility = View.VISIBLE

            val call = RetrofitFactory().retrofitService().getRCE(
                uf.text.toString(),
                cidade.text.toString(),
                rua.text.toString()
            )

            call.enqueue(object : Callback<List<CEP>> {

                override fun onResponse(call: Call<List<CEP>>?,
response: Response<List<CEP>>?) {

                    response?.body()?.let {
                        Log.i("CEP", it.toString())
                        Toast.makeText(this@MainActivity,
it.toString(), Toast.LENGTH_LONG).show()
                        progress_bar.visibility = View.INVISIBLE
                    } ?: Toast.makeText(
                        this@MainActivity,
                        "Endereço não localizado ",
                        Toast.LENGTH_LONG
                    ).show()
                }

                override fun onFailure(call: Call<List<CEP>>?, t:
Throwable?) {
                    Log.e("Erro", t?.message)
                    progress_bar.visibility = View.INVISIBLE
                }
            })
        }
    }
}
```

Código-fonte 7.5 – Classe MainActivity
Fonte: Elaborado pelo autor (2020)

Altere o conteúdo do arquivo chamado “*activity_main”* em xml.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/cep"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="Inserir                                    CEP"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"                />

    <Button
        android:id="@+id/pesquisaCEP"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="Pesquisar                com                CEP"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cep"           />

    <ProgressBar
        android:id="@+id/progress_bar"
        style="?android:attr/progressBarStyle"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
```

```
        android:layout_marginBottom="8dp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/pesquisaRCE"       />

    <EditText
        android:id="@+id/rua"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="Digite                    a                    RUA"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/pesquisaCEP"       />

    <EditText
        android:id="@+id/cidade"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="Digite                    a                    CIDADE"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/rua"               />

    <EditText
        android:id="@+id/uf"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
```

```
        android:layout_marginStart="8dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="Digite o ESTADO"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/cidade" />

    <Button
        android:id="@+id/pesquisaRCE"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="Pesquisar com Rua,Cidade e Estado"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/uf" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

Código-fonte 7.6 – Arquivo activity_main
Fonte: Elaborado pelo autor (2020)

Altere o conteúdo do arquivo chamado “AndroidManifest.xml”, adicionando as permissões necessárias para o uso de Internet entre a tag manifest como mostrado no “Código-fonte Arquivo Manifest.xml”.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.webservice">

    <!-- Permissoes para acessar a Internet -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
```

```
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Código-fonte 7.7 – Arquivo Manifest.xml
Fonte: Elaborado pelo autor (2018)

Execute o aplicativo no emulador, digite o seguinte CEP: 01538001 e veja os resultados da pesquisa.

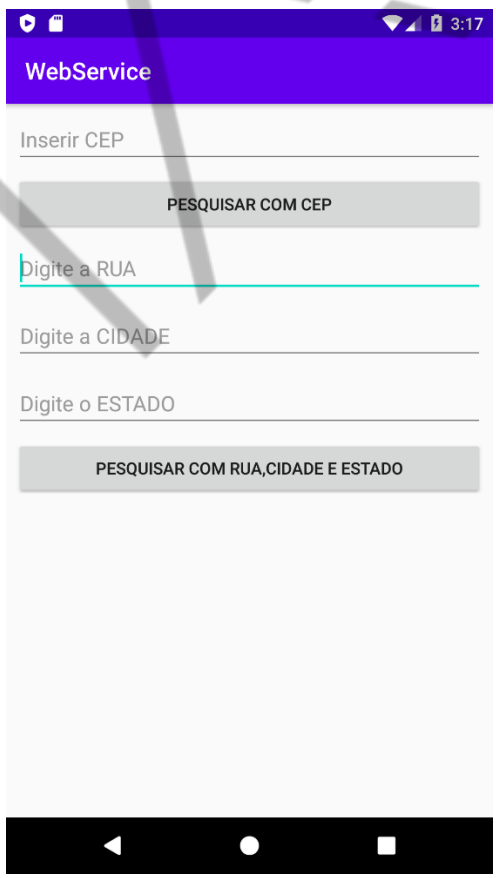

Figura 7.5 – Acessando webservice via App
Fonte: Elaborado pelo autor (2020)

7.2 Desenvolvendo notificações locais

Dentro do universo de comandos, funções e frameworks que orbitam a linguagem de programação Kotlin para o desenvolvimento de *Apps* para *Android,* existem classes muito importantes que transformam ideias em aplicativos. Tais classes ampliam a forma de comunicação com o usuário acionando toques sonoros, ativando a vibração para chamar a atenção e, principalmente, inserir notificações contendo informações importantes que precisam chamar a atenção do usuário.

Continuando nosso aprendizado, verificaremos como se desenvolvem rotinas que lembrem ao usuário sobre um novo compromisso ou uma atividade que precisa realizar por meio de notificações locais. Tais avisos podem acompanhar ícones personalizados, bem como um texto com diversas possibilidades de apresentação. Construa o exemplo a seguir e apresentaremos um conjunto de classes que permitem o desenvolvimento dessas notificações.

Construa um novo projeto para a plataforma *Android* com o *Android Studio*, nomeie esse projeto como “Notificação Local” e siga as configurações para um projeto padrão semelhante às orientações do projeto “*Webservices*”.

No projeto “Notificação Local”, você deverá executar os seguintes passos:

Crie três classes Kotlin com os seguintes nomes: *MainActivity.kt, TelaRespostaActivity.kt e Biblioteca.kt;*

Crie um arquivo do tipo “*Drawable Resource File*” com o seguinte nome: *ponteiro.xml;*

Insira uma permissão no arquivo “*Manifest.xml”* para liberação do uso do *VIBRATE;*

Passo 1: Crie uma classe Kotlin com o seguinte nome: *Biblioteca.kt.*

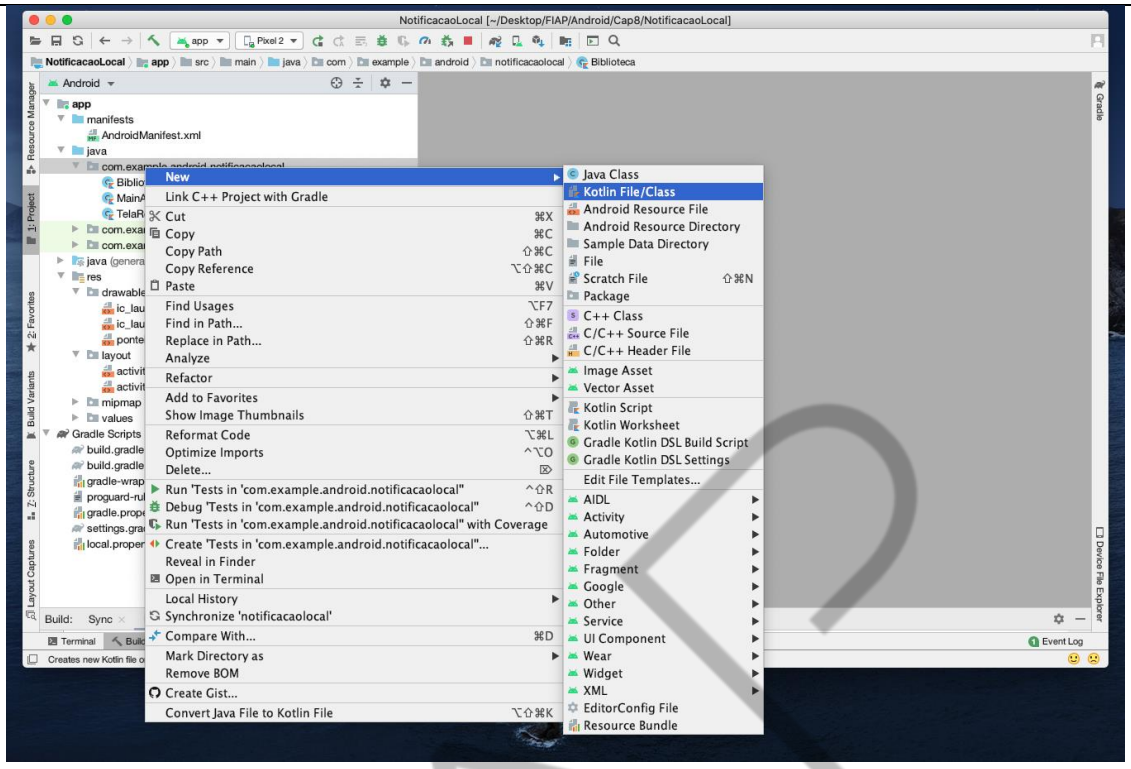

Figura 7.6 – Passo 1: Classe Biblioteca
Fonte: Elaborado pelo autor (2020)

Passo 2: Agora vamos criar uma nova Activity chamada TelaRespostaActivity e, para isso, vamos clicar com o botão direito no nosso projeto, opção New>Activity>Empty Activity, como mostra a Figura “Criando uma nova Activity”.

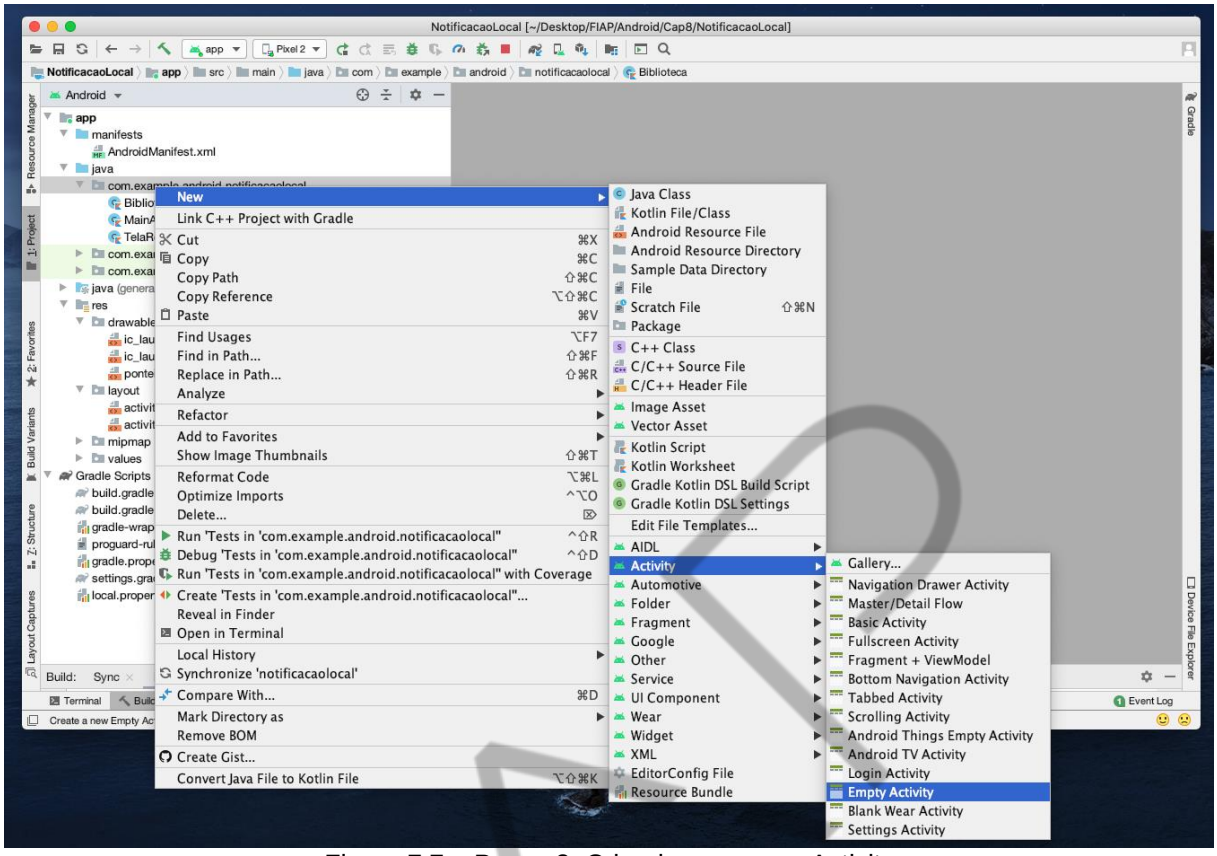

Figura 7.7 – Passo 2: Criando uma nova Activity
Fonte: Elaborado pelo autor (2020)

Abra o arquivo *MainActivity.kt* e troque as linhas de código por estas abaixo:

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.app.Notification
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.media.RingtoneManager
import android.util.Log
import android.view.View
import androidx.core.app.NotificationCompat

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    override fun onDestroy() {

        super.onDestroy()
        finish()
    }
```

```
    fun gerarNotificacaoFIAP(view: View?) {

        val notificationManager =
getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

        val p = PendingIntent.getActivity(this, 0,
                Intent(this, TelaRespostaActivity::class.java), 0)

        var mensagens = arrayOf("Reunião com o grupo",
                "Prova de matemática",
                "Apresentação do trabalho")

        val builder = NotificationCompat.Builder(this, "notification")

builder.setContentTitle(mensagens[(Math.random()*mensagens.size).toInt
()])
        builder.setSmallIcon(R.drawable.ponteiro)
        builder.setLargeIcon(Biblioteca.decoder())
        builder.setContentIntent(p)

        val style = NotificationCompat.InboxStyle()

        val unidades = arrayOf(
                arrayOf("FIAP","Campus Vila Olimpia","Rua Olimpíadas,
186","CEP: 04551-000"),
                arrayOf("FIAP","Campus Paulista","Av. Paulista,
1106","CEP: 01311-000"),
                arrayOf("FIAP","Campus Vila Mariana","Av. Lins de
Vasconcelos, 1264","CEP: 01538-001")
        )

        var random = (Math.random()*unidades.size).toInt()
        var unidade = unidades[random]
        for (detalhe in unidade) {
            style.addLine(detalhe)
        }
        builder.setStyle(style)

        val notificacao = builder.build()
        notificacao.flags = Notification.FLAG_AUTO_CANCEL


notificationManager.notify(R.drawable.ic_launcher_background,notificac
ao)

        try {

            val som =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION)
            val toque = RingtoneManager.getRingtone(this, som)
            toque.play()

        } catch (e: Exception) {

            Log.i("Erro", e.message.toString())
        }
    }
}
```

Código-fonte 7.8 – Arquivo MainActivity.kt
Fonte: Elaborado pelo autor (2018)

Abra o arquivo Biblioteca.kt e troque as linhas de código por estas abaixo:

```
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.util.Base64

class Biblioteca {

    companion object {

        val imagem64String: String =
"iVBORw0KGgoAAAANSUhEUgAAAKoAAACqCAMAAAAKqCSwAAAAolBMVEUfHx/tFFv///8hH
yAXHx0TIBvqFFpCHSnwFFwjHyD6E18eHx+wF0kPIBr9E2AUIBzbFVaDGjzCFk4oHyKRGUB
THC6aGUM6HifMFlEyHiWpGEcpKSlEHSr1FF2ysrLe3t5hYWFISEg2NjaMjIxNHSxnGzTJy
cmBgYGRkZFuGzbv7+9OTk7b29tAQEAvLy92GjhdHDG7F02JGT7fFVcHIRhhHDJAO6i8AAA
IqUlEQVR4nO2dCXeiOhTHcbFDWtG22pbO29GHcfv/9VegARCCC3Tpyzv8D/nvTMlL
D+Sy83NYmJ8K9Pm+/P28enBqE0PT4/b5++bUiBDf/jl9UeNkDngH68v1VE3b4/NYAo9vun
ytoi6ef3VLGikX69FWBX1/a0FoJF+vb1/jPrScNHLenz5CPW5aby8nktR37dNs6navutRN
y0qfKHHjQ71n6emuXT6+U8RddNKUsN42qio7y0s/USP7wrqtmmicm3zqC3zUnk9y6gvTdN
8rBcJ9WfTMB/rZ4b61jTLZ3oTqJuGQtPqethw1NemST7Xa4L63vpMZdn6HqO23lIjvcWoL
f/8E/2MUFvuU4VeGGoHPqpIrwz1R9MQ1fTjm9F+p5roYWN8b5qhqr4bHTFVZqzGtmmEqto
arY3+VT0aLW1SFfVkdMQBGN0B7dWrV69evXp1Q0B7MK/7PJlAjUicBKHmfOiRCjchd8AlV
qhREOOEYeF0ABe7oMBKwuEip9DwELkxLIDDCx4owvbKAwYBy2WBioS+O/cUCIDm1KSysL+
6Gupp/x2V+itVozGMUC8XFZVBuZieFIjo6Ggua7R03eXutqwM1V398VRFRkqA7yuowJvZk
729BMphhrr746FMnrGeu/R6U9YIdQQtoPmCNaiGsbRnf1bub5RjiFBn+ZsY0LlSvIA3ZI1
zVf/yRVTLGbtzRMLLYJhjiFFVo0hObgiVff2YWa+FDu75U1R2fXAplkptqCN3z/IJwLN7k
PNLj8pOo7e0gL9ABejKzBrE1wwuIZRTtKjeyh02g0oCf5DkEkB5MyzNVfM+uVqovxVUgKb
uGFk8iXmCjE2LCkiIVa92G1QiXGJasHlU4O3ohBDBNpOda4kH+K06tdugksVhH+uwEwRKr
pKJnVU/EdzYkf/iqJJfPdg3dQACFaC9yzUSPDlUyznK5gmYJ3JT8gyVl42DyHDlXtY3r63
iXA3311h7ba4CGOK8O/Bm8z1UUUk4jvV7zmKA8+IOMcDntho5nmvO8AB00jMFKoBrXjQsB
JrBW9aqRlUPANApMwxxpcZZkWAXax0CFrHcPl6t4FeJtRxoXCQh4k1SW+WB2T1aAZVQLfT
bPTpW4YQgSBxW0QPcGrMqKvCGzKVqLl1fzhYx9H61GVQDnm1dRB8716j+ag0qYEHfVFvti
KigNagslC6pdkSsVRuq/TEqcObuqaQuj92tw+q6uV1Prp7LUDEOCPCuZdkeV2IDZgJWLbn
K7O04Kwsq9nvmjODsGOp6WWIBuDuy8Jms9+EtI5MSEccrS0Ioer7nfEABvLh+hagGUubgS
0suceWg4Ptz51jZmb169erVq1evXr169ep1Y6kjgX+jT+/1ecpfdHFBJA+O/qWU3kGCHOU
EUvaUtO3psZRqWYr259FXdT5PdzLocDxSb8a7OuHxvMonzPmoB5mySyqhRuMg1P6KqOtOl
NY/CU4jN3c7d52gkvCSO05d3+KoS5ZQEXVs4y+JLk+FHgJmAOszlU4S48YA7fJXm0uBOmH
nV0V1C7MsqsheBUj3WXlwTM30jqYYubScoy09h72ohDq4KyodI6LtygCGc8Upa4oa9cRJr
DWi0qOTZmnsbCSPYzmzgVlAjcYTaPqk2lCxLY0EEuZtIGT/S2cAWc6VFlEBXGMzvUNNqDj
rkQUEBafpaDIZTU8WEsM/zK3wwpZQDQsdUhOoC9X0Qw4AoHH0mROi0X/LI8lKe5IUtoyaT
CeqFRXbBz7WDrzFxE19E3XPIc9t4M1oIVejzmX+BjWhsqeIXPKGF/mrxrYvxuGAN4pT8qj
R5BezTlT7yEcGSCB900mSmO4GvBMtojJz3SdX1INqXsJsuFK5FKcD8QTEr6GgxlM1cF2o2
Objwsz5mMVkcydq0rkto4r5GYRMouM1ofLy5zBq6lykXiVUAA+LZOiDmeuF1WU1GQAfwyS
BT4tXmnw0DsChZKuWMz9zy7DQidaUqyZOhoZY+WvTafYmpoQ6TYe+o2FwXA+qyLakhFVFT
pdbZ/RdZaj2IB1Rhme7DtT0GUCqJnOoIjyAkzxq9I78ewx902wxqstch6gf0MyswwAy1L0
e9ag3AGaf6VtED63DVnkNALydxq0O0tnNiYOQUdlrzjwejntn36ghXBEtu/gTL7KKNuoi5
6wiVJyGZMxcR0ENVcCeO/lcCyTN1DN39UkQkM9VbPPUqCaoA3UqUGfFKiD1VayytwuoLEZ
IJzSmNe0dUelStADgSM3WKFFGUFGZuZ5QvvV433hVTKlKqnM5aZClJR+diorNizK17L7x6
iqL9U2ZFQ9STwXQyi58VsnV6STY+6MyicmezJdfbBb9xwzMb5qiUk0DhAIqM9f8RKg7t63
SmYzAC1emTU0matPzGqXHuRkXUZkB7WVzvXfj+iAmCALoracTH2N/Oc9mgWa9PhpUFpvJ5
nrv3hW8FtUOABCRcLEISTYL1PJ2KZYGFdOJNFv83r0r1F8I1miGDYl+S5b2BFne4iI8rg4
1aoFlk7LunquMFZVMFLKcYdY80KJKPQk1dK9hejk5uq5Ai6ArzmoxPSoz13TG+P07LbFJp
```

```
wFS5v4CQJxgLnWvlqFmjYAaUCN/7x8CR/5NI/Gc4OjbWLpfCWpcj3wFdWB+SQPq+lPpN40
kmE19l+buRjPU/FMG7pHHtRN7YFZCNUi4Hn5Z69laRmV/qmeIWF/zFH7hgqVUQ9X/jrWqP
CgbAPtT/4NY7VN4ivTPT/VfRgPBZ3cqT5NTqqL2+r+qM+vZPHRpOaMOLRK1bRqhqrZdWtC
sQ8vEdWjxvQ4tadilhSI7tPxmhxY17dJSsR1agLcLH9Zr9xaLbr+1vnVxYfO2+1Z5ufgOL
cLfpa0NurRhRIe24Wjr5iZPms1NurRlTJc24mmfzyrf3uhblzaN6tJWXN86tMFZDNuVbeM
Sm33uxmZ8PG/btsXhvwHI38CsEBp+AAAAAElFTkSuQmCC"

        fun decoder(): Bitmap {

            val imageByteArray = Base64.decode(imagem64String,
Base64.DEFAULT)

            return
BitmapFactory.decodeByteArray(imageByteArray,0,imageByteArray.size)
        }
    }
}
```

Código-fonte 7.9 – Arquivo Biblioteca.kt
Fonte: Elaborado pelo autor (2018)

Abra o arquivo *TelaRespostaActivity.kt* e troque as linhas de código por estas:

```
import androidx.appcompat.app.AppCompatActivity
import android.content.Intent
import android.os.Bundle
import android.view.View
import kotlinx.android.synthetic.main.activity_tela_resposta.*

class TelaRespostaActivity : AppCompatActivity() {


    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_tela_resposta)

        image.setImageBitmap(Biblioteca.decoder())
    }

    fun retornar(view: View) {

        val intent = Intent(this, MainActivity::class.java)
        startActivity(intent)
    }

    override fun onDestroy() {
        super.onDestroy()
        finish()
    }
}
```

Código-fonte 7.10 – Arquivo TelaRespostaActivity.kt
Fonte: Elaborado pelo autor (2018)

Abra o arquivo *activity.xml* e troque as linhas de código por estas abaixo:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30dp"
        android:textStyle="bold"
        android:text="Notificação FIAP" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="gerarNotificacaoFIAP"
        android:text="Enviar" />

</LinearLayout>
```

Código-fonte 7.11 – Arquivo activity.xml
Fonte: Elaborado pelo autor (2018)

Abra o arquivo tela_resposta_activity.xml e troque as linhas de código por estas:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/image"
        android:layout_width="200dp"
        android:layout_height="200dp" />

    <Button
        android:onClick="retornar"
        android:text="Retornar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Código-fonte 7.12 – Arquivo tela_resposta.xml
Fonte: Elaborado pelo autor (2018)

Passo 3: Crie um arquivo do tipo “*Drawable Resource File*” com o seguinte nome: *ponteiro.xml.*

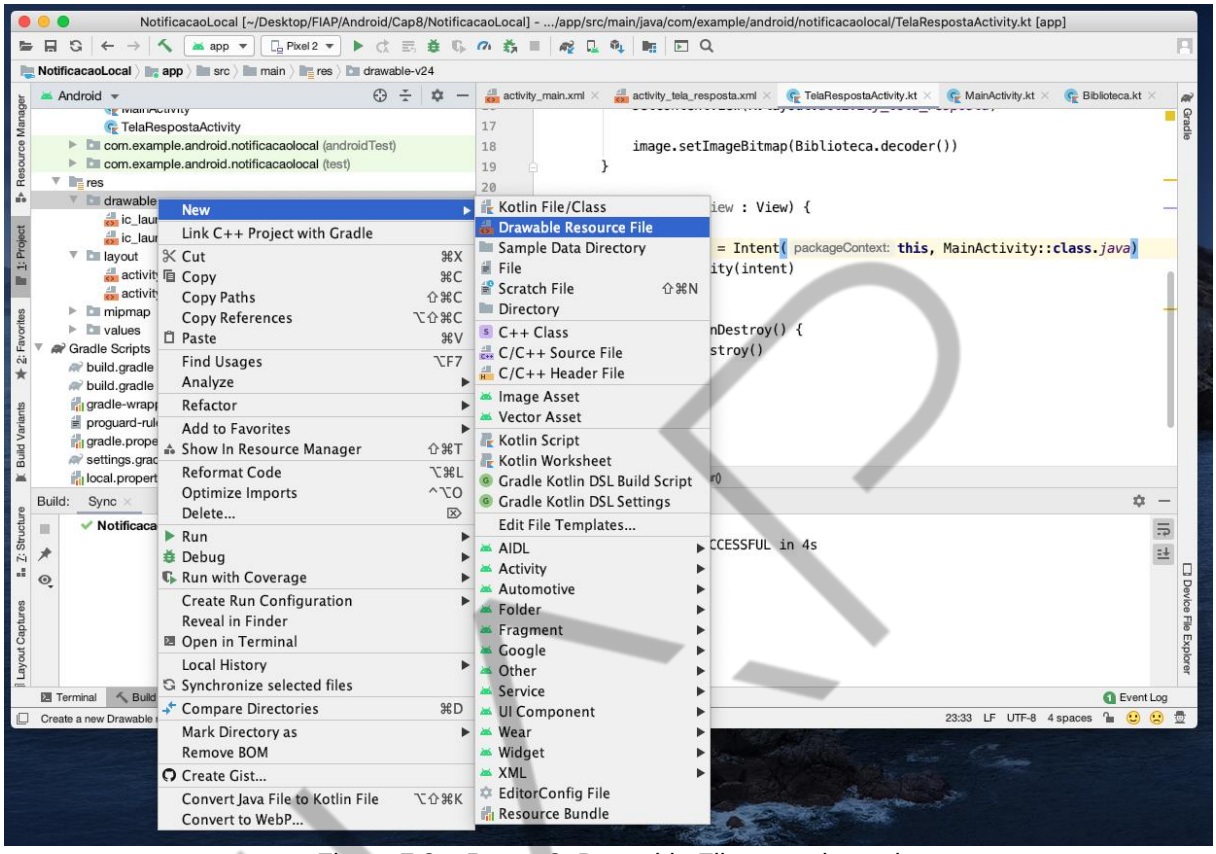

Figura 7.8 – Passo 3: Drawable File: ponteiro.xml
Fonte: Elaborado pelo autor (2020)

Abra o arquivo ponteiro.xml e troque as linhas de código por estas:

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="100dp"
    android:width="100dp"
    android:viewportHeight="100"
    android:viewportWidth="100">

    <path
        android:fillColor="@color/colorAccent"
        android:pathData="M 0,0 L 100,0 0,100 z" />

</vector>
```

Código-fonte 7.13 – Arquivo ponteiro.xml
Fonte: Elaborado elo autor (2018)

Passo 4: Inserir uma permissão uma *activity* no arquivo “*Manifest.xml”* para liberação do uso do *VIBRATE.*

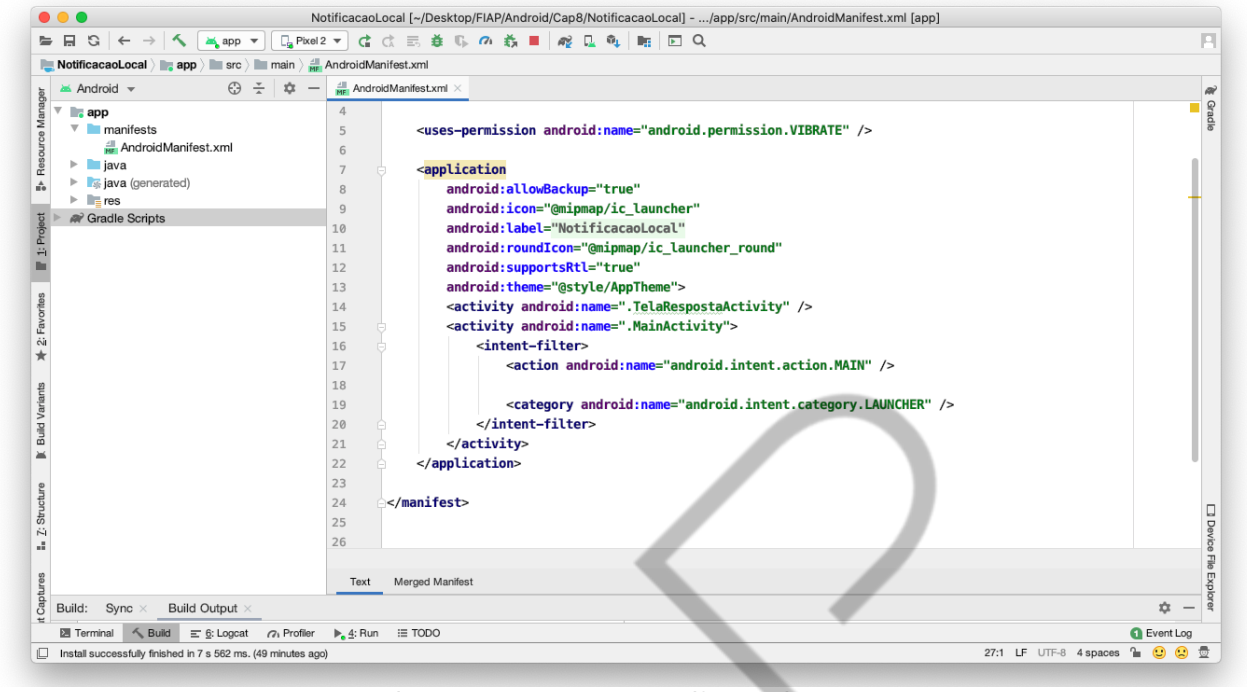

Figura 7.9 – Passo 4: Manifest.xml
Fonte: Elaborado pelo autor (2020)

```
<uses-permission android:name="android.permission.VIBRATE" />
```

Código-fonte 7.14 – Arquivo Manifest.xml
Fonte: Elaborado pelo autor (2018)

O aplicativo “Notificação Local” para Android ficará com as seguintes telas:

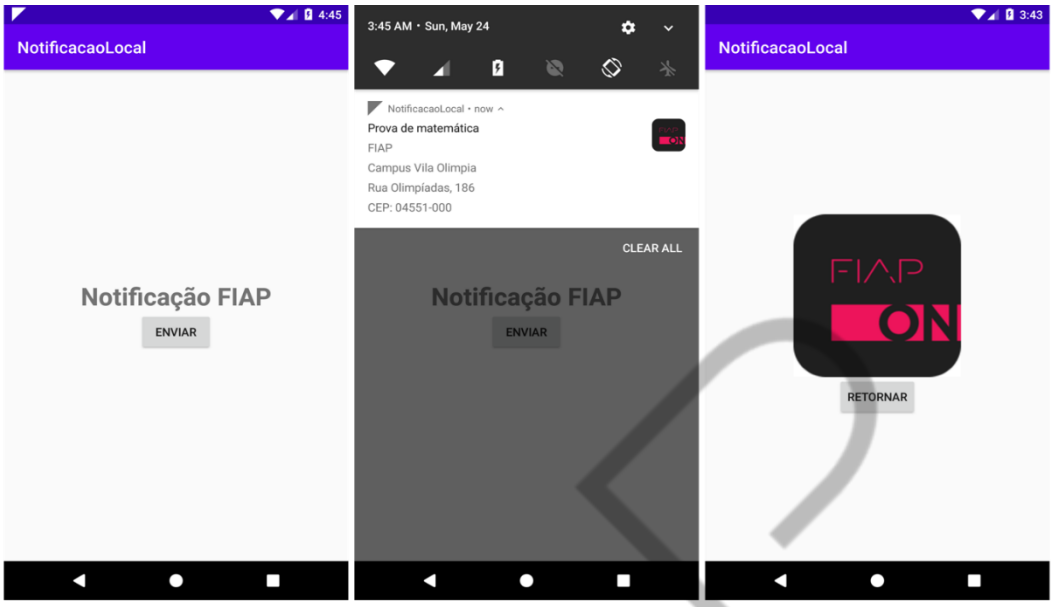

Figura 7.10 – Telas Finais do Aplicativo
Fonte: Elaborado pelo autor (2020)

7.3 Publicando App na Loja

Para publicar um aplicativo, é necessário seguir alguns passos importantes que vamos conhecer neste capítulo. Se você chegou até aqui, quer dizer que há um aplicativo pronto para publicar? Tenho certeza de que ainda não!. Acredito que alguns passos não foram feitos e agora vamos aprender e fazer juntos. Todo o processo de publicação é explicado detalhadamente na documentação oficial no link:<https://developer.android.com/studio/publish/?hl=pt-br>.

7.3.1 Controle de versão

Para publicar um aplicativo, precisamos controlar a versão do aplicativo que será publicado. Para defini-la, basta alterar os itens *versonCode* e *versionName* no arquivo *app/build.gradle*:

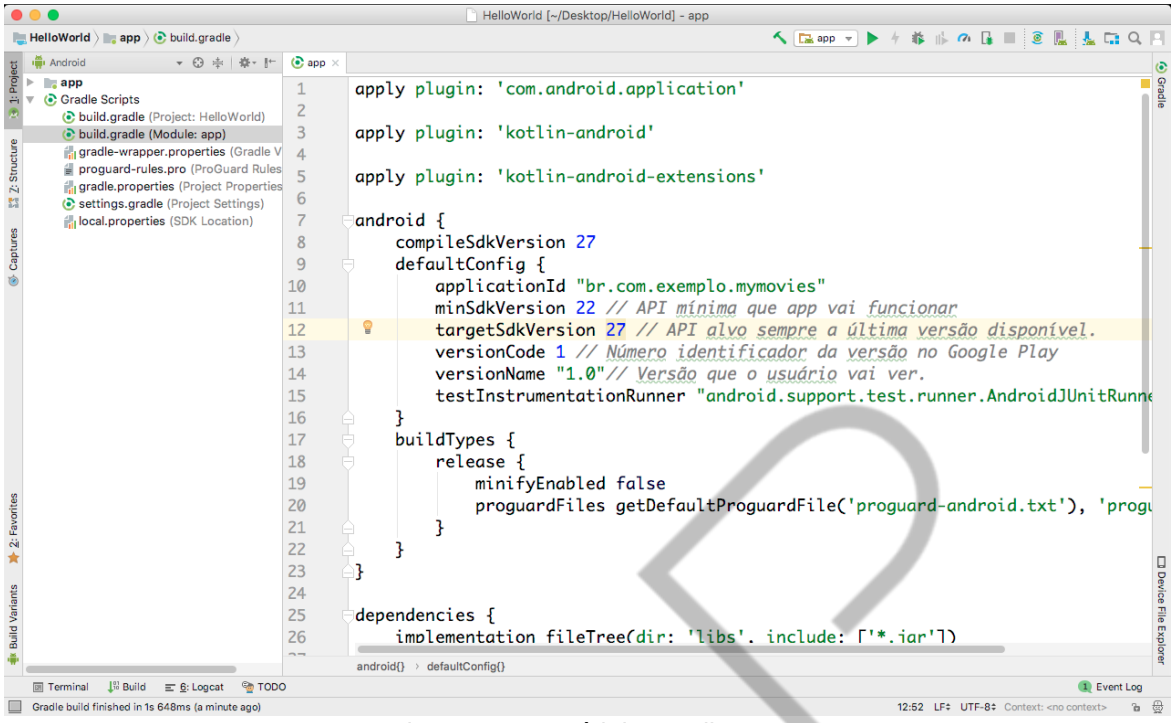

Figura 7.11 – Módulo Gradle App
Fonte: Elaborado pelo autor (2018)

```
...
android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "br.com.exemplo.mymovies"
        minSdkVersion 22 // API mínima que app vai funcionar
        targetSdkVersion 27 // API alvo sempre a última
versão disponível.
        versionCode 1 // Número identificador da versão no
Google Play
        versionName "1.0"// Versão que o usuário vai ver.
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
...
```

Código-fonte 7.15 – Configurando a versão no build.gradle (módulo app)
Fonte: Elaborado pelo autor (2018)

O item *versionCode* é o que o Google Play utiliza para controlar se o aplicativo teve atualização após a sua publicação na loja, por isso, é muito importante que esse número seja sempre maior que a versão publicada anteriormente. O item *versionName* é a versão que o usuário visualizará para saber em qual versão o aplicativo está.

7.3.2 Compilando o projeto

É fundamental configurarmos o item *minSdkVersion*, pois ele nos dirá qual a versão mínima em que o aplicativo funcionará. Além disso, é muito importante também configurar o *tragetSdkVersion* para a última versão lançada, para otimizar build e ativar os recursos dos novos dispositivos. O Google Play respeita essas configurações, disponibilizando o aplicativo apenas para as versões configuradas.

Para mais informações sobre versões, acesse: <https://developer.android.com/guide/topics/manifest/uses-sdk-element?hl=pt-br>.

7.3.3 Assinando o aplicativo

Para publicar um aplicativo, é necessário gerar um **apk** de *release*, podendo fazê-lo pelo menu **Build > Generate Signed APK** do *Android Studio*. Para gerar o **apk** de *release* e garantir a segurança, é preciso criar um certificado que o *Android Studio* o ajuda a fazer, basta seguir o passo a passo:

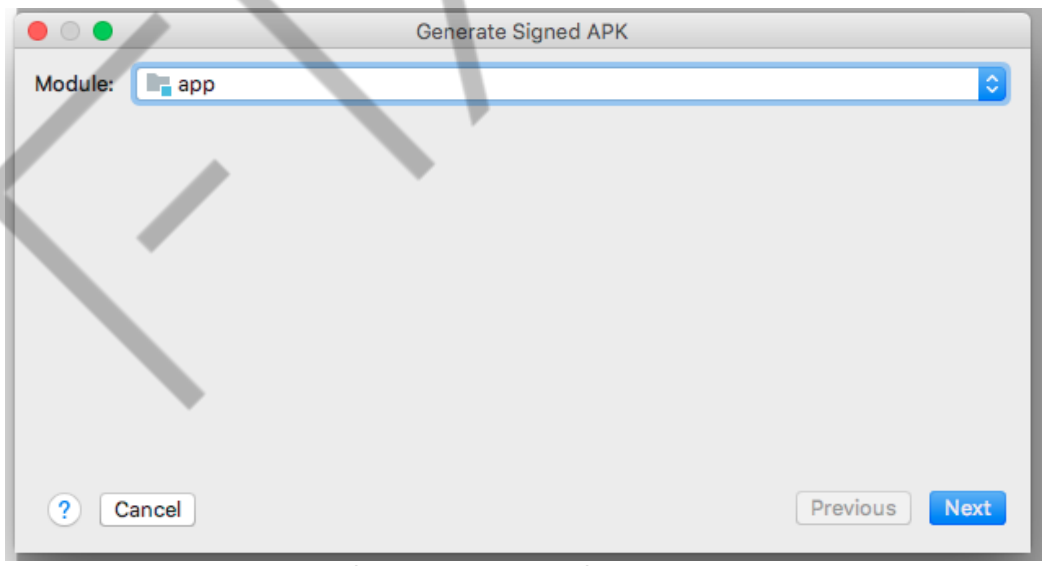

Figura 7.12 – Gerando um APK
Fonte: Elaborado pelo autor (2018)

Dica: Se houver dúvidas nesse processo, recomendamos ler a documentação oficial no link: <https://developer.android.com/studio/publish/app-signing?hl=pt-br>.

7.3.4 Publicando na Google Play

Para publicar um aplicativo na loja, é necessário possuir uma conta de desenvolvedor que tem um custo único de U$25 dólares. Para fazer esse cadastro e efetuar o pagamento, basta acessar o site do Google Play Console, lembrando que é preciso ter um cartão internacional para efetuar o pagamento.

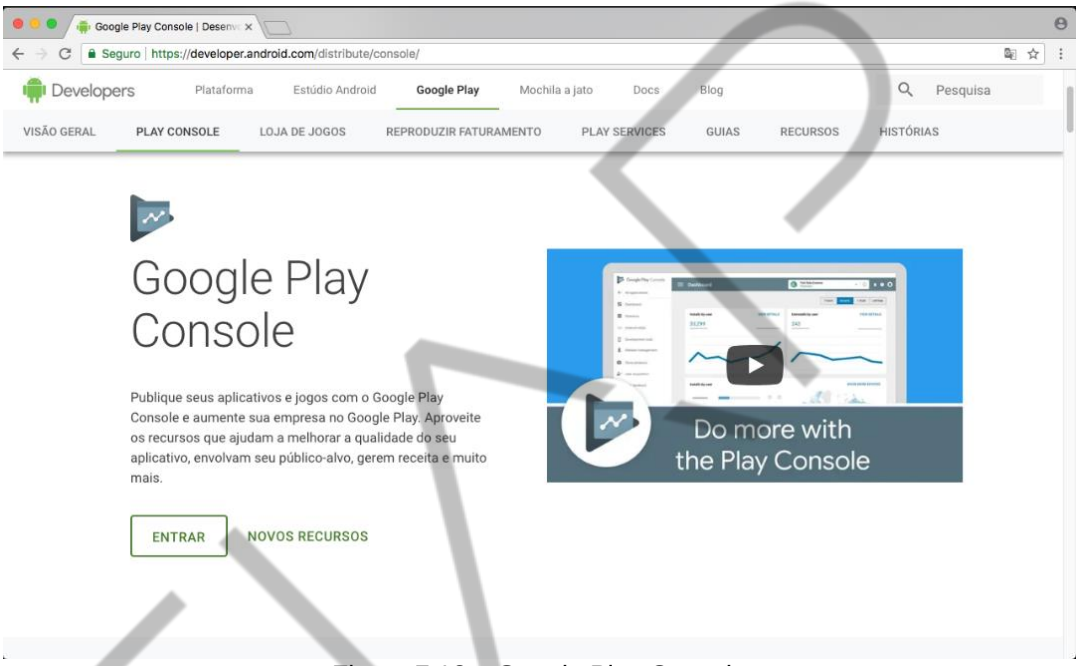

Figura 7.13 – Google Play Console
Fonte: Elaborado pelo autor (2018)

O processo de publicação é simples: após a conta ser liberada para utilização, basta acessar o console, clicar no botão **CRIAR APP**, preencher alguns campos do formulário que são obrigatórios, por exemplo, uma breve descrição do *App* e enviar fotos ou *screenshots* para divulgação. Adicione o **APK** com a assinatura necessária e publique. Para todo esse processo, há uma boa documentação no site oficial, explicando cada tópico ou campo do formulário que precisa ser preenchido.

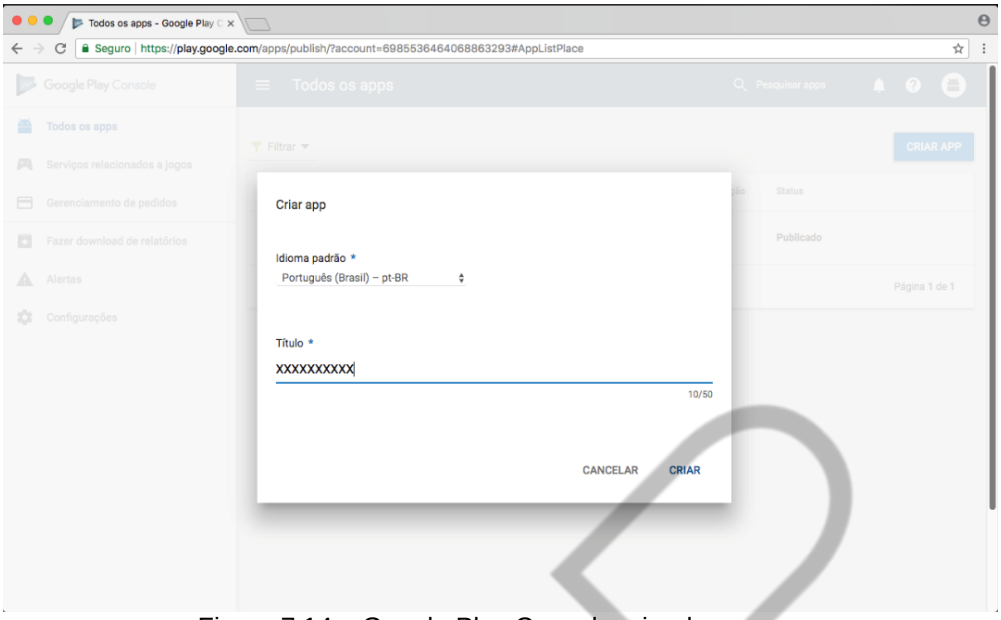

Figura 7.14 – Google Play Console criando um app
Fonte: Elaborado pelo autor (2018)

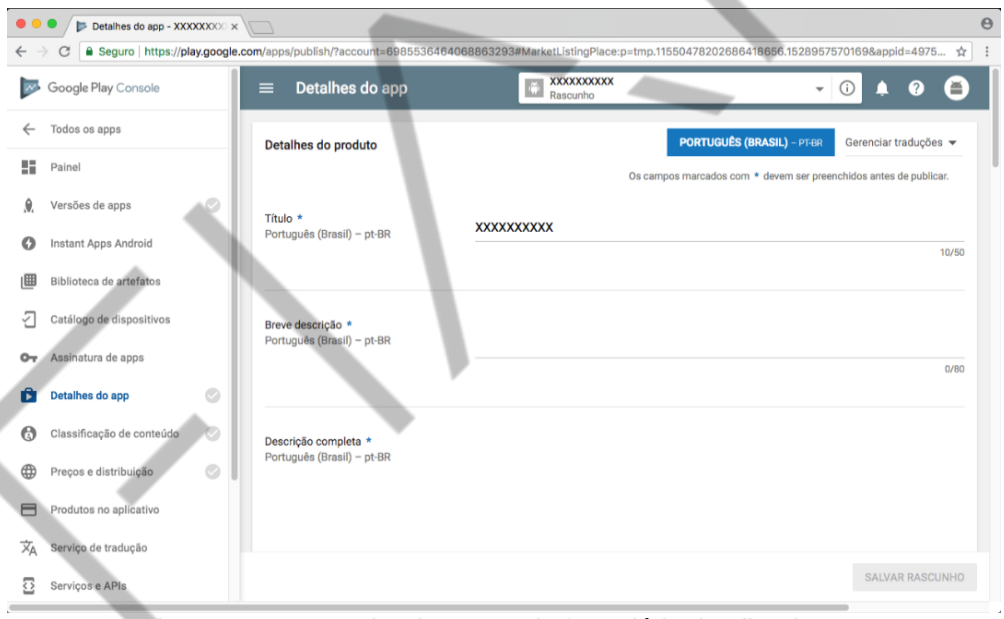

Figura 7.15 – Google Play Console formulário detalhe do app
Fonte: Elaborado pelo autor (2018)

Após completar os passos e publicar um aplicativo pelo console, será disponibilizado no Google Play depois de algumas horas e estará pronto para ser baixado e instalado pelos usuários. Lembre-se de alguns cuidados importantes a serem tomados: guarde bem o certificado de *release*, pois, sem ele, nunca mais será possível atualizar esse aplicativo, nunca se esqueça de incrementar a versão correta no *build.gradle* do projeto antes de gerar o **APK** e sempre acompanhe os comentários

e relatórios de erros enviados pelos usuários, para corrigir possíveis *bugs*, a fim de sempre melhorar e deixar o seu aplicativo funcionando.

REFERÊNCIAS

ANDROID. **Guia do Usuário**. 2020. Disponível em: <https://developer.android.com/?hl=pt-br>. Acesso em: 07 out. 2020.

LECHETA, Ricardo R. **Android Essencial com Kotlin**. São Paulo: Novatec, 2017.

RETROFIT. **A type-safe HTTP client**. 2020. Disponível em: <https://square.github.io/retrofit/>. Acesso em: 07 out. 2020.